Energy-Aware Scheduling of Conditional Task Graphs on NoC-Based MPSoCs

Umair Ullah Tariq*, Hui Wu* and Suhaimi Abd Ishak*†
*The University of New South Wales, Australia
†Universiti Tun Hussein Onn, Malaysia
{tariqu, huiw, sishak}@cse.unsw.edu.au

Abstract

We investigate the problem of scheduling a set of tasks with individual deadlines and conditional precedence constraints on a heterogeneous Network on Chip (NoC)-based Multi-Processor System-on-Chip (MPSoC) such that the total expected energy consumption of all the tasks is minimized, and propose a novel approach. Our approach consists of a scheduling heuristic for constructing a single unified schedule for all the tasks and assigning a frequency to each task and each communication assuming continuous frequencies, an Integer Linear Programming (ILP)-based algorithm and a polynomial time heuristic for assigning discrete frequencies and voltages to tasks and communications. We have performed experiments on 16 synthetic and 4 real-world benchmarks. The experimental results show that compared to the state-of-the-art approach, our approach using the ILP-based algorithm and our approach using the polynomial-time heuristic achieve average improvements of 31% and 20%, respectively, in terms of energy reduction.

1 Introduction

Modern mobile systems such as robots and driverless cars require computationally powerful and energy-efficient hardware due to their complex functions and battery power constraint. MPSoC is an ideal architecture for those mobile systems due to its high performance and low power dissipation. Examples of commercial MPSoCs include Samsung Exynos 5422 SoC [1] and Zynq UltraScale+ MPSoC devices [3]. Samsung Exynos 5422 SoC powers the famous Samsung Galaxy smartphone series. Zynq UltraScale+ MPSoC devices have been used in robots. Typically an MPSoC consists of processors with different power and performance profiles. For example, Samsung Exynos 5422 SoC consists of 4 high-performance ARM Cortex-A15 CPU, 4 low-power ARM Cortex-A7 CPU. Modern MPSoCs have a large number of processors and the number of processors on MPSoCs are expected to grow [10]. According to International Technology Roadmap for Semiconductors (ITRS), MPSoCs will integrate thousands of processors [15] by 2025. Therefore, the traditional bus-based on-chip communication is no longer feasible due to its poor scalability. NoC-based communication provides a significant improvement in terms of flexibility, scalability and performance over hierarchical (e.g., Advanced Microcontroller Bus Architecture and STBus) and traditional bus structures [23].

Mobile systems are battery powered. Although battery lifetimes have increased over the years, modern batteries are still far from meeting the needs of power-hungry mobile devices. Therefore, energy efficiency is a critical issue in mobile systems. One way to improve energy efficiency is to apply Dynamic Voltage and Frequency Scaling (DVFS). DVFS saves energy consumption by lowering the voltage/frequency of a processor/communication link when it is underutilized. For example, in order to reduce the energy consumption of a Nexus 4 Android smartphone on-demand governor scales the CPU frequency and voltage level based on CPU utilization every 50 millisecond [17]. In addition to processors, NoC communication links and routers also consume a large amount of on-chip energy. For Alpha 21364 processor [32], out of 125W total on-chip power consumption, 23W (20%) is consumed by NoC routers and links, and out of 23W, the NoC links consume 58% of the power. Therefore, it is important to take communication energy into account when mapping applications onto NoC-based MPSoCs.

In this paper, we target energy-efficient mobile embedded systems such as driver-less cars, robots and advanced combat helmets using a NoC-based MPSoC as the hardware platform. For those mobile systems, their complex functions such as object recognition and communication, are known at the design stage, and the embedded software is typically modelled as a set of tasks with conditional precedence constraints and individual deadlines. We investigate the problem of scheduling a set of tasks with conditional precedence constraints and individual deadlines on a heterogeneous NoC-based MPSoC such that the total expected processor and communication energy is minimized. The processors and NoC links are voltage scalable and can operate at a set of discrete voltage/frequency levels. We make the following major contributions:

1) We propose a novel offline task scheduling approach. Our approach consists of a task scheduling heuristic that constructs a single unified schedule for all the tasks and collectively assigns a frequency to each task and each communication assuming continuous frequencies, and an ILP-based algorithm and a polynomial-time heuristic for assigning a discrete frequency to each task and each communication. To the best of our knowledge, our approach is the first one that investigates the problem of scheduling a set of tasks and communications with

URI: http://hdl.handle.net/10125/50002
ISBN: 978-0-9981331-1-9

conditional precedence constraints on NoC-based MPSoCs such that the total expected energy consumption is minimized.

2) We have performed experiments on 20 benchmarks. Compared to the state-of-the-art approach proposed by Li and Wu [24] that does not consider conditional precedence constraints, our approach using the ILP-based algorithm achieves an average improvement of 31% and a maximum improvement of 61%, and our approach using the polynomial-time heuristic achieves an average improvement of 20% and a maximum improvement of 46%. Furthermore, both our approach using the ILP-based algorithm and our approach using the polynomial-time heuristic run approximately three times faster than the state-of-the-art approach.

The rest of this paper is organized as follows. Section 2 gives an overview of the related work. Section 3 describes all the models, including the task model, the power models, and the MPSoC model. Section 4 presents our heuristic for task scheduling and frequency assignment assuming continuous frequencies for both processors and communications. Section 5 proposes an ILP-based algorithm and a polynomial-time heuristic for assigning discrete frequencies to tasks and communications. Section 6 presents our experimental results and analysis. Lastly, Section 7 concludes this paper.

2 Related Work

Several approaches have been proposed to minimize energy consumption for heterogeneous multi-processors systems. Gebotys et al. [12] investigate the problem of scheduling tasks onto heterogeneous processors such that total energy consumption is minimized. In their approach task mapping and scheduling are integrated with dynamic voltage scaling to maximize energy efficiency. Singh et al. [38] propose a contention-aware, energy efficient, duplication-based mixed integer programming (CEEDMIP) formulation for scheduling task graphs on NoC-based heterogeneous multiprocessors. The key idea of their approach is to duplicate some tasks to reduce the communication energy as well as traffic congestion. Zhang et al. [45] propose an ILP-based, energy-aware task mapping algorithm on heterogeneous multi-processors, and an evolutionary algorithm-based, energy-efficient task mapping heuristic. Cai et al. [6] propose an energy efficient approach for heterogeneous multi-processor mobile embedded systems. Their approach assigns discrete frequencies to tasks based on the critical path lengths of tasks. Lin et al. [25] propose an energy-efficient algorithm for heterogeneous MPSoC-based mobile devices. They integrate task mapping and scheduling with dynamic voltage scaling to reduce the energy consumption of mobile devices.

Huang et al. [21] propose a simulated annealing-based energy-aware task mapping algorithm on heterogeneous NoC-based MPSoCs. In their model, processors are assumed to be voltage scalable and NoC links operate at a fixed frequency. Mixed Integer Linear Programming (MILP) is used to assign voltages/frequencies to tasks. Shin et al. [37] consider a NoC model with voltage scalable links and propose a genetic algorithm for minimizing the communication energy of the NoC by scaling the link voltages. Ghosh et al. [13] consider a model similar to that of Huang et al. [21] and propose an energy-aware task scheduling heuristic based on MILP relaxation and randomized rounding. Li et al. [24] assume a NoC model with voltage scalable links. They propose a task mapping algorithm and a genetic algorithm-based task voltage/frequency assignment algorithm. A detailed survey on approaches for multi-processor energy-efficient embedded computing is given in [31].

Only a few approaches have been proposed aiming at minimizing the energy consumption of tasks with conditional precedence constraints. Shin et al. [36] propose a scenario-based offline Non-Linear Programming (NLP) algorithm that assigns each task a speed for each scenario. The approach has an exponential time complexity as it constructs a separate schedule for each scenario. Wu et al. [41] propose an approach that employs the schedule table generated by [9] to identify the available slack time in the worst-case and propose a heuristic that assigns a frequency to each task. In [28] a heuristic is proposed to assign each task a speed based on the critical path length. The heuristic has an exponential time complexity in the worst case since it enumerates all the possible scenarios when computing the critical path lengths. Umair et al. [40] propose a task mapping algorithm for periodic CTGs, and propose an NLP-based algorithm and a heuristic to assign voltages/frequencies to tasks.

Energy efficiency is not only critical in mobile embedded systems but also important in cloud computing. In cloud data centers, efficient power management may reduce operation costs, increase system reliability and reduce adverse effects of large power consumption on environments. Many approaches have been proposed to minimize energy consumption in the data center. Hasan et al [19] formulate the problem of offline scheduling of jobs on the servers of a data center such that the total energy consumption is minimized, as a binary integer program. They also propose an online heuristic for the same problem. Sarood et al. [35] propose an Integer Linear Programming (ILP)-based scheduler to reduce energy consumption in data centers. Huai et al. [20] propose a load balancing algorithm combined with DVFS to significantly reduce the energy consumption of data center servers. Roukh et al. [34] argue that database management systems (DBMSs) are one of the major energy consumers in data centers, and propose a machine learning-based approach to reduce the energy consumption of nodes of database clusters when optimizing queries. Xu et al. [42] propose an energy-aware query optimization platform called PET for DBMSs. PET estimates the energy costs of queries offline and the evaluation engine of the DBMS configures PET parameters towards a desired energy/performance trade-off. Guo et al. [16] propose an energy efficient query processing framework in DBMSs. Their approach works out energy cost query plans and makes a trade-off between the performance and the energy plans. Authors in [18] and [14] discuss practices in detail to reduce the energy footprint of data centers. Mittal et al. [30] give a survey of power management techniques for data centers.

Our approach differs from all the previous approaches in three major aspects. First, our approach considers conditional

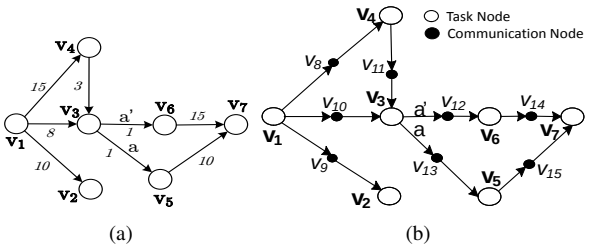


Fig. 1. (a) A CTG G (b) An extended CTG G_e

precedence constraints. Second, our approach handles NoC and takes link contentions into account. Third, our approach collectively optimizes the frequencies of processors and NoC links aiming at minimizing the total expected energy consumption of the MPSoC.

3 Models

The target application is modelled by a conditional task graph (CTG) [39]. A CTG is a weighted directed acyclic graph $G(V, E, A, X)$ defined as follows. $V = \{v_1, v_2, \ldots, v_n\}$ is a set of tasks. Each task has an execution time represented by the number of clock cycles on each processor and a deadline d_i. All the tasks are non-preemptible. $E \subseteq V \times V$ is a set of directed edges each denoting the dependency between the two tasks. A is a set of triplets $(e_i, c_i, p(c_i))$, where $e_i \in E$, and c_i and $p(c_i)$ represent the condition associated with e_i and its probability [26], respectively. X is a set of edge weights. An edge weight $\chi_s \in X$ of an edge $e_s = (v_i, v_j)$ represents the communication volume in bits from task v_i to task v_j. Our task model is described in detail in [39]. The execution probability of each node $v_i \in V$ is represented by $p(v_i)$. We use algorithm presented in [26] to compute the probabilities.

The MPSoC has a set $P = \{pe_1, pe_2, \ldots, pe_m\}$ of m processors. We assume heterogeneous processors, where each processor $pe_k \in P$ is DVFS-enabled and can operate on a set $\{(V_{dd_1}, f_1), \ldots, (V_{dd_{n_k}}, f_{n_k})\}$ of n_k discrete voltage-frequency pairs. A matrix NC represents the execution times in clock cycles of all the tasks in G on different processors, where $NC(j, i)$ is the number of clock cycles of task v_i on pe_j.

The dynamic power $P_{d_{k,i}}$ of a task v_i on processor pe_k, dominated by discharging and charging of load capacitance due to gate switching, is given as $P_{d_{k,i}} = C_{eff_{k,i}} V_{dd_{k,i}}^2 f_{k,i}$ [5], [4], where $C_{eff_{k,i}}$, $V_{dd_{k,i}}$ and $f_{k,i}$ are the effective load switching capacitance, the supply voltage and the operating frequency, respectively. The execution time of a task v_i on processor pe_k operating at frequency $f_{k,i}$ is given as $t_{k,i} = NC(k, i)/f_{k,i}$. The operating frequency f is approximated by $f = ((1 + K_1)V_{dd} + K_2 V_{bs} - V_{th_1})^\alpha / K_6 L_d V_{dd}$ [4], where K_1, K_2, K_6 and V_{th_1} are circuit dependent constants, L_d is the logic depth, and α is velocity saturation imposed by the used technology $(1.4 \leq \alpha \leq 2)$. The total energy consumption $E_{k,i}$ of a task v_i on pe_k is computed as follows [4]:

$$E_{k,i} = NC(k,i) C_{eff_{k,i}} V_{dd_{k,i}}^2 + L_g (V_{dd_{k,i}} K_3 e^{K_4 V_{dd_{k,i}}} e^{K_5 V_{bs}} + |V_{bs}| I_j) t_{k,i} \quad (1)$$

We consider the 2D mesh NoC architecture, where each processor is associated with a router, and there are N_R rows and N_C columns. Every router has five ports with one port used to communicate with the associated processor, and the remaining four ports used to communicate with the neighboring routers. A link connecting two routers is called global link and a link connecting a router with its associated processor is referred to as a local link. All the links are full duplex. All the global links are identical and have same link width (also called bus width or the number of wires) b_w.

We only take into account the energy consumption of global links and neglect the energy consumption of the local links. In the rest of this paper, links refer to global links unless they are explicitly specified.

The NoC links can operate at a set $\{(V_{dd_1}, f_1), \ldots, (V_{dd_F}, f_F)\}$, of voltage-frequency pairs. In a 2D mesh, the Manhattan distance $\eta_{i,j}$ between two processors pe_i and pe_j is defined as follows: $\eta_{i,j} = |x_i - x_j| + |y_i - y_j|$, where (x_i, y_i) and (x_j, y_j) are the coordinates of pe_i and pe_j, respectively.

The wormhole switching [27], [24] and deterministic XY routing are used. We do not scale router frequencies as adjusting router frequencies makes the problem too complex. We assume that router frequencies are fixed and commensurate with link frequencies as in [24].

Consider the message e_i for a communication node. The time taken by e_i on the links operating at frequency f_i such that e_i traverses the network without contention is calculated as follows [24]:

$$t_i = \frac{\chi_i}{b_w f_i} \quad (2)$$

We use the bit energy model given in [43], [29] for communication. Assume that the source node and the destination node of e_i are mapped on processors pe_s and pe_d, respectively. The energy of transmitting one bit of the message e_i is $E_{bit} = (\eta_{s,d} + 1) E_{Rbit} + \eta_{s,d} E_{lbit_i}$, where E_{Rbit} is the energy consumption of one bit on one router, and E_{lbit_i} is the energy consumption of transmitting one bit on one link when all the links of e_i operate at f_i. Thus, the energy consumption of transmitting e_i on the links operating at frequency f_i is calculated as follows:

$$E_{comm_i} = \chi_i ((\eta_{s,d} + 1) E_{Rbit} + \eta_{s,d} P_i / (f_i b_w)) \quad (3)$$

where P_i is the total power consumed in transmitting one bit when the links that e_i traverses operate at frequency f_i. P_i is the sum of the dynamic power P_{dyn_i} and static power P_{stat_i}, $P_i = P_{dyn_i} + P_{stat_i}$ [4]. The static and dynamic powers depend on how links are implemented. The frequency f_i is approximated by $f_i = ((1 + K_1)V_{dd} + K_2 V_{bs} - V_{th_1})^\alpha / K_6 L_d V_{dd}$ [4].

4 Task Mapping, Scheduling and Frequency Assignment

In order to schedule tasks and communications in a unified way, we first transform a CTG G into an extended CTG by adding an additional node for every edge in the original CTG. We refer to these additional nodes as communication nodes. The original nodes in G are kept unchanged and are referred to as task nodes. Specifically, for each edge $(v_i, v_j) \in G$, we add

a communication node v_s, and replace (v_i, v_j) by two directed edges (v_i, v_s) and (v_s, v_j). If $(v_i, v_j) \in G$ has a condition, (v_i, v_s) has the same condition and probability. The extended graph is represented by $G_e(V + V^*, E', A')$, where V is a set of task nodes, V^* is a set of communication nodes, E' is a set of edges, and A' is a set of 3-tuples where each 3-tuple consists of an edge, the condition associated with the edge and probability of the condition. Figure 1(b) shows the extended graph G_e of the CTG in Figure 1(a).

4.1 Successor-Tree-Consistent Deadline

Our offline scheduling algorithm schedules nodes using the priorities of task nodes and communication nodes. We extend the notion of successor-tree-consistent deadline [39] to NoC-based MPSoCs, and propose a priority scheme for nodes, where the priority of each node v_i is its successor-tree-consistent deadline denoted by d'_i. When computing the successor-tree-consistent deadline of each node, we assume that all the processors and NoC-links operate at the maximum frequencies. Furthermore, the original CTG is used rather than the extended graph G_e. Before defining the successor-tree-consistent deadline, we introduce the worst case set of a task. Let $IPred(v_i)$ and $ISucc(v_i)$ be the sets of all the immediate predecessors and all the immediate successors of a task v_i, respectively.

Definition 1: The worst-case set of a task v_i, denoted by $WCS(v_i)$, is a set of tasks defined as follows:

1) If v_i is a sink node, $WCS(v_i) = \emptyset$.
2) If v_i is an OR-FORK node, $WCS(v_i) = \{v_j\} \cup WCS(v_j)$, where v_j is in $ISucc(v_i)$ satisfying $d'_j - \min_{\forall pe_k \in P} \{t_{k,j}\} = \min_{\forall v_s \in ISucc(v_i)}\{d'_s - \min_{\forall pe_k \in P}\{t_{k,s}\}\}$.
3) If v_i is an AND-FORK node, $WCS(v_i) = \bigcup_{v_s \in ISucc(v_i)} (WCS(v_s) \cup \{v_s\})$.

Definition 2: Given a CTG G and a task v_i, the successor tree of a task v_i is a weighted directed tree ST(G,v_i)= (V', E', X') where $v' = \{v_i\} \cup WCS(v_i)$, $E' = \{(v_i, v_j) : v_j \in WCS(v_i)\}$, and $X' = \{w'_{i,j}$: if v_j is an immediate successor of v_i, $w'_{i,j}$ is equal to the edge weight of (v_i, v_j) in G; otherwise, $w'_{i,j} = 0\}$.

Definition 3: Given a task v_i, if v_i is a sink task, its successor-tree-consistent deadline d'_i is equal to its preassigned deadline d_i. Otherwise, d'_i is the upper bound on the latest completion time of v_i in any feasible schedule of the relaxed problem instance: a set $V' = \{v_i\} \cup WCS(v_i)$ of tasks with the precedence constraints in the form of the successor tree $ST(G, v_i)$, where the deadline of each task $v_j \in WCS(v_i)$ is its successor-tree-consistent deadline, and the deadline of v_i is its preassigned deadline, and the same MPSoC.

The successor-tree-consistent deadlines of all the tasks in G are computed as follows. For each task v_i in reverse topological order of G, if v_i is a sink node, its successor-tree-consistent deadline d'_i is equal to its preassigned deadline d_i and $WCS(v_i)$ is an empty set. Otherwise, compute $WCS(v_i)$, construct the successor tree of v_i, and do the following:

1) Partition the tasks in $WCS(v_i)$ into two disjoint sets U and J. The set U consists of all the tasks in $WCS(v_i)$ each of which does not receive any data from v_i, and the set J contains all the tasks in $WCS(v_i)$ that are not in U.
2) Sort all the tasks in U in non-increasing order of their successor-tree-consistent deadlines.
3) Schedule each task v_j in U on a processor that maximizes its start time.
4) Sort all the tasks in J in non-increasing order of their successor-tree-consistent deadlines. For the tasks with the same successor-tree-consistent deadlines, sort them in non-increasing order of their edge weights.
5) Schedule each task v_j in J on a processor that maximizes its start time.
6) Schedule v_i on a processor that maximizes its completion time respecting the constraints specified by the successor tree of v_i.
7) Set d'_i to the completion time of v_i.

A communication node v_s in the extended graph G_e has a single child node v_j. Therefore, the successor-tree-consistent deadline of v_s is $d'_s = d'_j - \min_{\forall pe_k \in P}\{t_{k,j}\}$, where the execution time $t_{k,j}$ of a task node v_j is computed assuming the maximum processor frequency.

4.2 Earliest Successor-Tree-Consistent Deadline First Algorithm

In a CTG, the number of scenarios grows exponentially as the number of conditions increases. Therefore, it is not feasible to construct a separate schedule for each scenario. Our offline scheduling approach constructs a single unified schedule for all the scenarios by exploiting the mutual exclusion relations between communication and task nodes. Two nodes are said to be mutually exclusive in the graph G_e if they cannot co-exist in any scenario. For example, in Figure 1(a) v_5 and v_6 are mutually exclusive. Two mutually exclusive nodes can be allocated the same resource at the same time. In a CTG, two nodes are said to be concurrent if they are not reachable from each other in graph G_e and are not mutually exclusive.

We propose an Earliest-Successor-Tree-consistent Deadline First (ESTDF) list scheduling algorithm assuming that all processors and links operate at the maximum frequencies. ESTDF is called by our main algorithm IOETCS described in the next subsection. It determines the order in which task nodes and communication nodes execute and captures this order by adding additional precedence constraints in the input graph G. The output of ESTDF is the input graph G with the additional precedence constraints. Given a CTG G, a matrix NC of worst-case clock cycles of tasks, a vector X of communication volumes and a task-to-processor mapping Map, ESTDF works by constructing a set $ReadySet$ containing the source nodes of G and repeating the following steps until $ReadySet$ is empty.

1) Select a node v_j with the minimum successor-tree-consistent deadline from $ReadySet$.
2) Compute its ready time $r_j = \max\{\zeta_l : v_l \in IPred(v_j)\}$, where ζ_l is the finish time of the node v_l.
3) If v_j is a communication node, compute its finish time $\zeta_j = r_j + t_j$, where t_j is given in Equation (2), and insert unconditional directed edges in G from v_j to the communication nodes that are concurrent to v_j, have

larger or equal successor tree consistent deadlines as compared to v_j and traverse the same links that v_j traverses.

4) If v_j is a task node, compute its finish time $\zeta_j = r_j + t_{k,j}$, and insert unconditional directed edges from v_j to unscheduled nodes concurrent to v_j and mapped on the same processor where v_j is mapped.
5) Delete v_j from $ReadySet$ and insert all ready nodes in G to $ReadySet$.

Consider the CTG in Figure 1(b) and the MPSoC in Figure 2(a) where all the processors are identical. The execution times of tasks at the maximum processor frequency are $t_{1,1} = 7, t_{1,2} = 2, t_{1,3} = 5, t_{1,4} = 3, t_{1,5} = 3, t_{1,6} = 2, t_{1,7} = 4$ time units. The communication times are $t_8 = 7, t_9 = 8, t_{10} = 6, t_{11} = 5, t_{12} = 4, t_{13} = 5, t_{14} = 7, t_{15} = 9$ time units. All the tasks have a common deadline of 40 time units. Consider the task mapping in Figure 2(a). Based on this task mapping the input CTG G_e shown in Figure 1(b) to ESTDF does not contain communication nodes v_{11}, v_{14} and v_{15}. Furthermore edges (v_4, v_{11}), (v_{11}, v_3), (v_6, v_{14}), (v_{14}, v_7), (v_5, v_{15}), (v_{15}, v_7) in G_e are replaced by (v_4, v_3), (v_6, v_7) and (v_5, v_7). Figure 2(b) gives an illustration of ESTDF scheduling algorithm for task mapping in Figure 2(a). Three communication nodes v_8, v_{10} and v_9 become ready after v_1 is scheduled. Communication nodes v_8 and v_{10} traverse the same link l_1. Since they are concurrent, they contend for l_1. ESTDF resolves this conflict by scheduling v_8 before v_{10} as v_8 has a smaller successor-tree-consistent deadline than v_{10}. Since v_8 and v_{10} are concurrent nodes, an edge is inserted from v_8 to v_{10} to capture this order as shown in Figure 2(c). Notice that communication nodes v_{12} and v_{13} are allocated the same time slot even though both use the same link l_3. This is because both are mutually exclusive. No additional edges are inserted between v_{12} and v_{13} as they are not concurrent nodes.

4.3 Iterative Offline Energy-Aware Task and Communication Scheduling Algorithm (IOETCS)

We propose an iterative offline energy-aware task and communication scheduling algorithm (IOETCS), Algorithm 1, for a NoC-based MPSoC. IOETCS constructs a single unified schedule iteratively assuming continuous frequencies for both processors and links.

IOETCS repeats three major steps until all the nodes in G_e are mapped and scheduled. First, it selects an unscheduled task node $v_i \in V$ with the smallest successor-tree-consistent deadline among all the unscheduled task nodes. Second, it initializes the initial energy consumption E_{ini} of the schedule to infinity and repeats the following steps for every $pe_k \in P$:

1) Tentatively assign v_i to the processor pe_k by $Map[i] \leftarrow k$ and construct a sub-graph $G_s(V_s + V_s^*, E_s)$ where V_s is the set of all the mapped task nodes, V_s^* is a set of communication nodes with both child and parent nodes mapped on different processors and E_s is a set of all the edges where every edge in E_s belongs to E' and both its head and tail nodes are in $V_s + V_s^*$. For each communication node v_s whose parent node v_p and child node v_c are mapped on the same processor, insert a directed edge (v_p, v_c) to E_s.

ALGORITHM 1: IOETCS

input : CTG $G_e(V + V^*, E', A')$ with a matrix NC and a set X, node deadlines, and a NoC-based MPSoC
output: Schedule graph $G^*(V_s + V_s^*, E_s)$, a vector Map for task mapping, and a communication and task voltage assignment.

```
Construct a list L of nodes in V sorted in non-descending order of successor-tree-consistent deadlines;
∀v_i ∈ VMap[i] ← 0;
for each v_i ∈ L in order do
    E_ini ← ∞; p ← 0;
    for each pe_k ∈ P do
        Map[i] ← k;
        Construct graph G_s;
        G_s ← ESTDF(G_s, NC, X, Map);
        Compute voltage assignment of nodes in G_s and total expected energy E_exp of G_s by solving NLP;
        if E_exp < E_ini then
            G* ← G_s; E_ini ← E_exp; p ← k;
    Map[i] ← p;
```

2) Call $G_s \leftarrow ESTDF(G_s, NC, X, Map)$ to construct a local schedule and capture the resource constraints introduced by the local schedule.
3) Given a task-to-processor mapping and a graph G_s, assign voltages/frequencies to task and communication nodes by solving a non-linear programming (NLP) problem. The objective of the NLP is to minimize the total expected energy consumption of graph G_s. The expected energy consumption is given as $E_{exp} = \sum_{v_i \in V_s} p(v_i) E_{k,i} + \sum_{v_i \in V_s^*} p(v_i) E_{comm_i}$. The NLP problem is formulated as follows:

$$\min\{E_{exp}\}$$

Subject To

$$\forall v_i \in V_s \quad t_{k,i} = \frac{NC(k,i) K_6 L_d V_{dd_{k,i}}}{((1+K_1)V_{dd_{k,i}} + K_2 V_{bs} - V_{th_1})^{\alpha}} \tag{4}$$

$$\forall v_i \in V_s^* \quad t_i = \frac{\chi_i K_6 L_d V_{dd_i}}{b_w((K V_{dd_i} + K_2 V_{bs} - V_{th_1})^{\alpha})} \tag{5}$$

$$\forall v_i \in V_s \qquad \rho_i + t_{k,i} \le d'_i \tag{6}$$

$$\forall (v_i, v_j) \in E_s \wedge v_i \in V_s \qquad \rho_i + t_{k,i} \le \rho_j \tag{7}$$

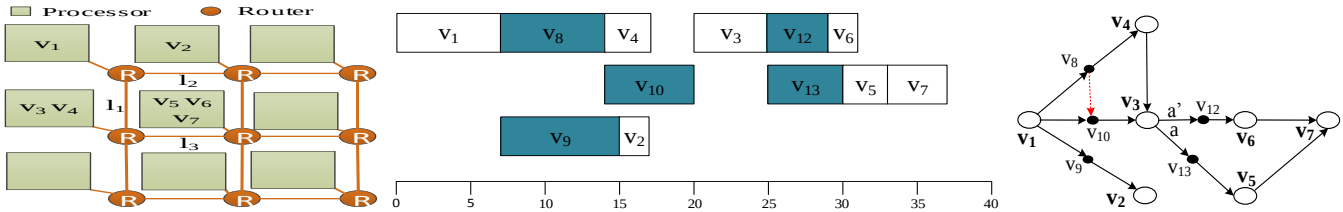
$$\forall (v_i, v_j) \in E_s \wedge v_i \in V_s^* \qquad \rho_i + t_i \le \rho_j \tag{8}$$

$$\rho_i \ge 0 \tag{9}$$

$$V_{dd_{k,min}} \le V_{dd_{k,i}} \le V_{dd_{k,max}} \tag{10}$$

$$V_{dd_{min}} \le V_{dd_i} \le V_{dd_{max}} \tag{11}$$

In Equation (5), $K = K_1 + 1$. The decision variables are, the start time ρ_i, the task node execution time $t_{k,i}$, the communication time t_i, the task voltage $V_{dd_{k,i}}$ and the communication voltage V_{dd_i}. $V_{dd_{k,max}}$ and $V_{dd_{k,min}}$ are



(a) Mapping (b) Local Schedules (c) Schedule Graph

Fig. 2. An illustrative example (a) Task-to-processor mapping (b) Local schedules constructed by ESTDF (b) Graph capturing the precedence and resource constraints

the minimum and the maximum supply voltages of the processor pe_k, respectively. Equations (4) and (5) are the task execution time and communication time constraints, respectively. Equation (6) is the deadline constraint, and the Equations (7), (8) are precedence constraints. Since the constraints and the objective function are convex, this NLP problem can be solved in polynomial time [44].

4) If the initial energy E_{ini} is greater than E_{exp}, set $p \leftarrow k$, $G^* \leftarrow G_s$, and $E_{exp} \leftarrow E_{ini}$.

In the final step, IOETCS maps v_i to processor p, and set $Map[i] \leftarrow p$.

5 Discrete Frequency Assignment

Algorithm 1 constructs a graph $G^*(V_s + V_s^*, E_s)$ that captures the original precedence constraints and constraints introduced by the schedule, and assigns an optimal frequency/voltage to each node in G^*. However, the frequency/voltage level assigned to a node may not be a valid discrete frequency/voltage of the processor/link where the node is mapped. Therefore, we propose an ILP-based algorithm and a polynomial time heuristic for assigning a discrete frequency to each node.

5.1 ILP-Based Algorithm

The optimal frequency f_i^{opt} of a communication node and the optimal frequency $f_{k,i}^{opt}$ of a task node are computed as described in Section 4. We differentiate between the following two cases for each task or communication node v_i:

1) If v_i is a task node and its frequency $f_{k,i}^{opt}$ is a discrete frequency of the processor pe_k where v_i is assigned, assign $f_{k,i}^{opt}$ to v_i. If v_i is a communication node and its frequency f_i^{opt} is equal to a discrete link frequency, assign f_i^{opt} to v_i.
2) If v_i is a task node and its frequency $f_{k,i}^{opt}$ is not a discrete frequency of the processor pe_k where v_i is assigned, find two frequencies $f_{k,i}^{opt,u}$ and $f_{k,i}^{opt,l}$ of the pe_k where v_i is assigned such that $f_{k,i}^{opt,u}$ is the smallest discrete frequency of pe_k larger than $f_{k,i}^{opt}$ and $f_{k,i}^{opt,l}$ is the largest discrete frequency of pe_k smaller than $f_{k,i}^{opt}$. Similarly, if v_i is a communication node and its frequency f_i^{opt} is not a discrete link frequency, find two discrete frequencies $f_i^{opt,l}$ and $f_i^{opt,u}$ of communication links such that $f_i^{opt,u}$ is the smallest discrete frequency of communication links larger than f_i^{opt} and $f_i^{opt,l}$ is the largest discrete frequency of communication links smaller than f_i^{opt}. Clearly, the optimal discrete frequency of v_i must be either $f_i^{opt,u}$ or $f_i^{opt,l}$ for a communication node and either $f_{k,i}^{opt,u}$ or $f_{k,i}^{opt,l}$ for a task node.

We introduce a binary decision variable to select between $f_i^{opt,u}$ and $f_i^{opt,l}$ if v_i is a communication node or between $f_{k,i}^{opt,u}$ and $f_{k,i}^{opt,l}$ if v_i is a task node.

$$x_i = \begin{cases} 0 & \text{if } v_i \text{ uses } f_i^{opt,l} or f_{k,i}^{opt,l} \\ 1 & \text{if } v_i \text{ uses } f_i^{opt,u} or f_{k,i}^{opt,u} \end{cases}$$

Let V^{opt} be a set of nodes that lie in Case 1. $V_R = V_s \setminus V^{opt}$ is a set of task nodes and $V_R^* = V_s^* \setminus V^{opt}$ is a set of communication nodes for which Case 2 holds. The expected energy consumption is now given as $E_{exp} = \sum_{v_i \in V_R}((1-x_i)E_{k,i}^{opt,l} + x_i E_{k,i}^{opt,u})p(v_i) + \sum_{v_i \in V_R^*}((1-x_i)E_{comm_i}^{opt,l} + x_i E_{comm_i}^{opt,u})p(v_i) + C$, where $E_{k,i}^{opt,l}$ and $E_{k,i}^{opt,u}$ (given in Equation (1)) are the energy consumptions of a task node v_i on a processor pe_k at the frequencies $f_{k,i}^{opt,l}$ and $f_{k,i}^{opt,u}$, respectively, $E_{comm_i}^{opt,l}$ and $E_{comm_i}^{opt,u}$ (given in Equation (3)) are the energy consumptions of a communication node v_i when all the links on its routing path operate at the frequencies $f_i^{opt,l}$ and $f_i^{opt,u}$, respectively and C is the sum of energy consumption of nodes in V^{opt}. The ILP problem is formulated as follows:

$$\min\{E_{exp}\}$$

Subject To

$$\forall v_i \in V_R \quad t_{k,i} = t_{k,i}^{opt,l}(1-x_i) + t_{k,i}^{opt,u} x_i \tag{12}$$

$$\forall v_i \in V_R^* \quad t_i = t_i^{opt,l}(1-x_i) + t_i^{opt,u} x_i \tag{13}$$

$$\forall v_i \in V_R \qquad \rho_i + t_{k,i} \leq d_i' \tag{14}$$

$$\forall v_i \in V^{opt}, V_s \qquad \rho_i + t_{k,i}^{opt} \leq d_i' \tag{15}$$

$$\forall (v_i, v_j) \in E_s \wedge v_i \in V^{opt}, V_s \qquad \rho(v_i) + t_{k,i}^{opt} \leq \rho_j \tag{16}$$

$$\forall (v_i, v_j) \in E_s \wedge v_i \in V^{opt}, V_s^* \qquad \rho(v_i) + t_i^{opt} \leq \rho_j \tag{17}$$

$$\forall (v_i, v_j) \in E_s \wedge v_i \in V_R \qquad \rho(v_i) + t_{k,i} \leq \rho_j \tag{18}$$

$$\forall (v_i, v_j) \in E_s \wedge v_i \in V_R^* \qquad \rho(v_i) + t_i \leq \rho_j \tag{19}$$

$$\rho_i \geq 0 \tag{20}$$

The decision variables are task execution time $t_{k,i}$, communication time t_i, binary variable x_i and start time ρ_i. $t_{k,i}^{opt,l}$ and $t_{k,i}^{opt,u}$ are the execution times of the task node v_i on the processor pe_k where v_i is mapped at the frequencies $f_{k,l}^{opt}$ and $f_{k,u}^{opt}$, respectively. $t_i^{opt,l}$ and $t_i^{opt,u}$ are the communication times (given in Equation (2)) of the communication node

v_i when all the links of the communication path operate at the frequencies $f_i^{opt,l}$ and $f_i^{opt,u}$, respectively. $t_{k,i}^{opt}$ is the execution time of the task node v_i at frequency level $f_{k,i}^{opt}$ and t_i^{opt} is the communication time of the communication node v_i when all the links of the communication operate at the frequency f_i^{opt}. Equation (12) defines the execution time of a task node. Equation (13) defines the communication time of a communication node. Equations (14) and (15) collectively define the deadline constraints, Equations (16), (17), (18) and (19) collectively define the precedence constraints.

5.2 Heuristic Algorithm

The ILP problem is a well-known NP-Complete problem. Therefore, the previous ILP-based algorithm is not scalable. Next, we propose a polynomial time heuristic to assign discrete frequencies to task and communication nodes. The heuristic uses the schedule constructed by IOETCS algorithm (Algorithm 1) and works as follows:

1) Compute the cuts of graph G^* as follows:
   - Create a copy G' of G^* and repeat the following steps until G' is empty:
     a) Create a cut containing all the source nodes with zero in-degree in G'.
     b) Remove all the source nodes and their incident edges from G'.
2) For every node $v_i \in V_s + V_s^*$, if its optimal frequency computed by NLP is a discrete frequency, assign the optimal frequency to v_i. Otherwise, assign $f_{k,i}^{opt,l}$ to v_i if v_i is a task node or $f_i^{opt,l}$ to v_i if v_i is a communication node.
3) Construct a new local schedule using the new frequency such that the order between nodes remain the same as in the schedule used by the NLP-based algorithm.
4) If there is no late task node, the algorithm terminates. Otherwise, repeat the following steps until there is no late task node.
   - Find the first late task node v_j and repeat the following steps until v_j is not late.
     a) Find a set B of nodes where every node $v_z \in B$ satisfies the following two conditions. First, v_z belongs to the set $\{v_j\} \cup Pred(v_j)$, where $Pred(v_j)$ is a set of predecessors of v_j. Second, the frequency of v_z has not been adjusted before and v_z has not been assigned an optimal discrete frequency by NLP.
     b) For every node $v_i \in B$, compute its rank. The rank of v_i is a 2-tuple (g_i, κ_i) which reflects the impact of v_i on shifting the late node v_j to an earlier time. Let C_p be a set of nodes of a cut containing v_i, C'_p be $C_p \cap B$, FT_j^{old} the finish time of v_j in the current schedule, FT_j^{new} the finish time of v_j after the frequencies of all the nodes in the set C'_p are increased by one level, and $FT_j^{new,i}$ the finish time of v_j when the frequency of v_i is increased by one level. The normalized time gain g_i of the cut containing v_i, is given as $g_i = \frac{FT_j^{old} - FT_j^{new}}{E_T + E_C}$, where $E_T = \sum_{v_i \in C'_p, V_s} (E_{k,i}^{opt,u} - E_{k,i}^{opt,l}) p(v_i)$ and $E_C = \sum_{v_i \in C'_p, V_s^*} (E_i^{opt,u} - E_i^{opt,l}) p(v_i)$. The normalized time gain κ_i of v_i is computed as:

$$\kappa_i = \begin{cases} \frac{FT_j^{old} - FT_j^{new,i}}{(E_i^{opt,u} - E_i^{opt,l}) p(v_i)} & v_i \ is \ commun- \\ & ication \ node \\ \frac{FT_j^{old} - FT_j^{new,i}}{(E_{k,i}^{opt,u} - E_{k,i}^{opt,l}) p(v_i)} & v_i \ is \ task \ node \end{cases}$$

     c) Select a node with the highest rank by comparing ranks lexicographically. Adjust its frequency to $f_i^{opt,u}$ if v_i is a communication node or $f_{k,i}^{opt,u}$ if v_i is a task node. Update the schedule.

TABLE I
CHARACTERISTICS OF BENCHMARKS WITHOUT CONDITIONAL PRECEDENCE CONSTRAINTS.

BM	$a/b/D$	Dim	BM	$a/b/D$	Dim
TG 1	17/19/1.4	4x5	TG 2	20/24/0.77	5x4
TG 2	15/11/0.98	4x5	TG 4	16/12/0.89	5x4
TG 5	27/28/2.4	6x5	TG 6	27/35/2.7	6x5
TG 7	27/39/2.9	6x5	TG 8	30/40/3.45	6x6

TABLE II
CHARACTERISTICS OF BENCHMARKS WITH CONDITIONAL PRECEDENCE CONSTRAINTS.

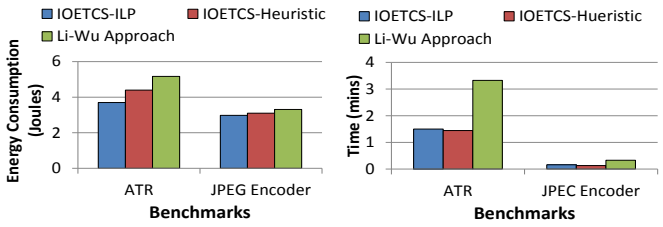
BM	$x/y/z/D$	Dim	BM	$x/y/z/D$	Dim
CTG1	17/2/6/0.74	3x3	CTG2	20/1/2/1.06	3x3
CTG3	15/2/4/0.723	3x3	CTG4	17/2/6/0.93	3x3
CTG5	30/4/11/1.73	3x3	CTG6	35/3/8/3.101	3x2
CTG7	33/5/15/3.7128	3x2	CTG8	31/3/9/3.69	3x2

6 PERFORMANCE EVALUATION

In this section, we use IOETCS-ILP and IOETCS-Heuristic to denote our approach using the ILP-based algorithm and the heuristic, respectively, for assigning a discrete frequency to each task and each communication. To demonstrate the effectiveness of IOETCS-ILP and IOETCS-Heuristic, we compare them with three approaches. The first approach is Li-Wu approach, a state-of-art approach for unconditional task graph model proposed in [24]. The second approach ILP-vpv-flv that is the same as IOETCS-ILP except that the NLP and ILP algorithms are modified such that they only scale processors frequencies/voltages and assign the maximum link frequency to all communication nodes. The third approach is ILP-fpv-vlv that is the same as IOETCS-ILP except the NLP and ILP algorithms are modified such that they only scale the voltages of links and assign the maximum processor frequencies to task nodes.

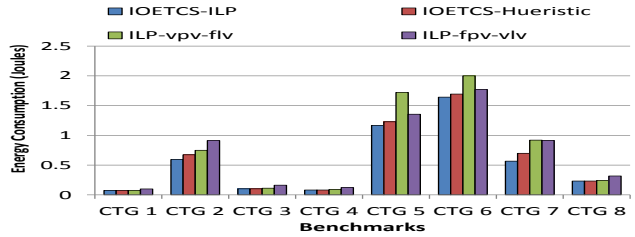
6.1 Simulation Setup

We use the same experimental setup as in [11], [7], [4]. The technology parameters are taken from [7]. We use two types of processors in our experiments, Type 1 and Type 2, modelled after the processors in [7] and [8], respectively. The configuration for NoC links are adopted from [24]. The execution times in cycles of tasks are randomly generated within $[10, 100] \times 10^6$ and $[5, 10] \times 10^6$, respectively.

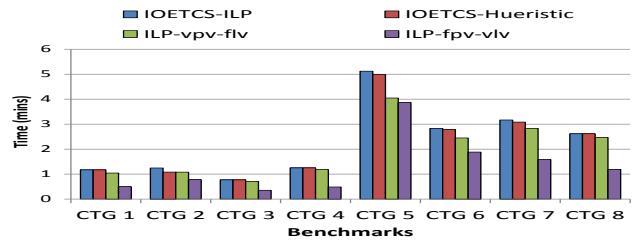


(a) Energy Consumption (b) Running Time

Fig. 3. Comparison of real-world benchmarks without conditional precedence constraints

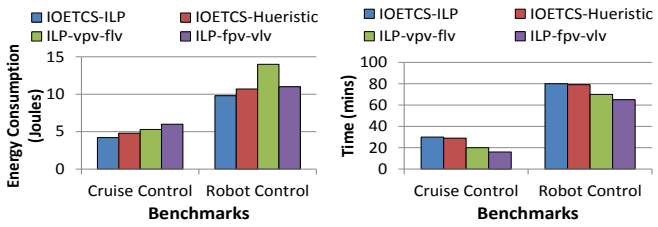


(a) Energy consumption



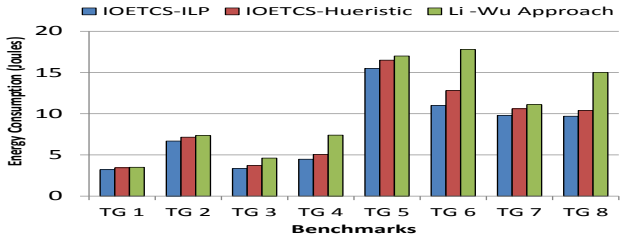
(b) Running time

Fig. 4. Comparison of eight benchmarks in Table II

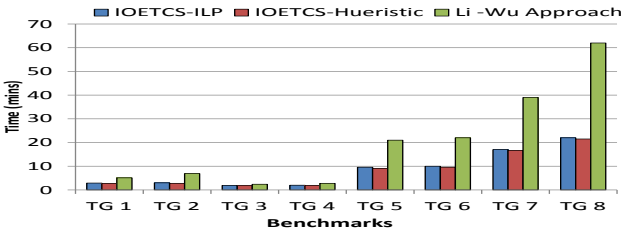


(a) Energy consumption (b) Running time

Fig. 5. Comparison of real-world benchmarks with conditional precedence constraints



(a) Energy consumption



(b) Running time

Fig. 6. Comparison of eight benchmarks in Table I

The communication volumes are generated randomly within $[80, 800] \times 10^6$ in bits. The deadline for each application is set to twice the makespan of the schedule of the application constructed by IOETCS algorithm assuming the maximum processors frequencies, the maximum links frequencies and a common deadline of 300 seconds for all the tasks so that there is reasonable slack for energy reduction. All the approaches are implemented in Matlab version R2015a. We use fmincon, quadprog and intlinprog solvers to solve the NLP, quadratic programming and ILP problems, respectively. The hardware platform consists of Intel(R) Core(TM) i5-4570 CPU with a clock frequency of 3.20 GHz, 8.00 GB memory, and 3 MB caches.

6.2 Results and Discussion

6.2.1 Experiments with conditional task graphs: In the first set of experiments we choose eight benchmarks and their details are given in Table II where x/y/z/D stands for the number of tasks, the number of OR-FORK tasks, the number of conditions and the deadline of the application in seconds, respectively. The column with heading Dim represents NoC dimensions. The benchmarks in Table II are the same benchmarks used in [26].

IOETCS-ILP achieves an average improvement of 31%, a maximum improvement of 62 % for CTG 7 and a minimum improvement of 1.03 % for CTG 1 over ILP-vpv-flv. It achieves an average improvement of 27%, a maximum improvement of 61% for CTG 3 and a minimum improvement of 7.9% for CTG 6 over ILP-fpv-vlv. IOETCS-Heuristic achieves an average improvement of 23%, a maximum improvement of 40% for CTG 5 and a minimum improvement of 1.3% for CTG 1 in comparison to ILP-vpv-flv. It achieves an average improvement of 18%, a maximum improvement of 61% for CTG 3 and a minimum improvement of 4% for CTG 6 over ILP-fpv-vlv. We observe that ILP-vpv-flv performs significantly better in terms of energy consumption if the computation energy dominates the total energy, and ILP-fpv-vlv performs better if communication energy dominates the total energy. CTG 5, 6 and 7 favour ILP-fpv-vlv as the communication volumes for these benchmarks are significantly larger than the execution times of task nodes. Both IOETCS-ILP and IOETCS-Heuristic distribute slacks efficiently between communication nodes and task nodes and thus perform significantly better than ILP-vpv-flv and ILP-fpv-vlv. In terms of running time both ILP-vpv-flv and ILP-fpv-vlv run slightly faster than IOETCS-ILP and IOETCS-Heuristic. This is because the search space of ILP-vpv-flv and ILP-fpv-vlv is smaller as compared to IOETCS-ILP and IOETCS-Heuristic. ILP-vpv-flv only scales processor voltages and ILP-fpv-vlv only scales link voltages. Whereas, IOETCS-ILP and IOETCS-Heuristic scale both the processor

voltages and the link voltages.

We choose two real-world benchmarks vehicle cruise controller [33] and Robot control [2] that are the task graphs of actual applications. These benchmarks are executed on 3x3 NoC where the processors are selected randomly as either Type 1 or Type 2. Both IOETCS-ILP and IOETCS-Heuristic perform significantly better than ILP-vpv-flv and ILP-fpv-vlv in terms of energy consumption. In terms of running time IOETCS-ILP and IOETCS-Heuristic take longer time compared to ILP-vpv-flv and ILP-fpv-vlv. The reason is that IOETCS algorithm cannot find a feasible solution for some sub-problems, and thus the solver takes a longer time to converge.

6.2.2 Experiments with non-conditional task graphs: To demonstrate the effectiveness of our approach on task graphs without conditional precedence constraints, we have conducted a second set of experiments. We choose eight task graphs (TG) and their details are given in Table I where a/b/D stand for the number of tasks, the number of edges and the deadline of the application in seconds, respectively. The column with the heading Dim represents NoC dimensions. The benchmarks in Table II are the same benchmarks used in [26] except that all the edges are treated as unconditional edges. Figure 6(a) gives a comparison of 8 benchmarks in Table I in terms of energy consumption where all the processors are of Type 1. IOETCS-ILP achieves an average improvement of 31%, a maximum improvement of 61% for TG 6 and a minimum improvement of 9% for TG 1 over Li-Wu approach. IOETCS-Heuristic achieves an average improvement of 20%, a maximum improvement of 46% for TG 4 and a minimum improvement of 2% for TG 1 over Li-Wu approach. We observe that Li-Wu approach makes very poor mapping decisions for heterogeneous processors. The benchmarks TG 3, TG 4, TG 6 and TG 8 are executed on MPSoCs where the processors are randomly selected as either Type 1 or Type 2. The reason for poor performance of Li-Wu approach is that it does not take into account the energy profiles of processors when making mapping decisions. The benchmarks TG 1, TG 2, TG 5 and TG 8 are executed on MPSoCs with homogeneous processors (Type 1). As a result, Li-Wu approach performs considerably better. In terms of running time, IOETCS-ILP and IOETCS-Heuristic run approximately three times faster than Li-Wu approach. The major reason is that the genetic algorithm takes significantly longer time as it constructs a new schedule for each candidate solution using ETFGBF.

We have chosen two real-world benchmarks JPEG encoder [22] and Automatic Target Recognition (ATR) [24]. JPEG encoder is executed on a 3x3 MPSoC and ATR is executed on a 4x5 MPSoC. The processors are randomly selected as either Type 1 or Type 2. For both benchmarks, IOETCS-ILP and IOETCS-Heuristic outperform Li-Wu approach in terms of both running time and energy consumption.

We observe that the energy consumption of the schedules produced by IOETCS-Heuristic are close to those of the schedules produced by IOETCS-ILP. IOETCS-ILP achieves the average improvement of 11% over IOETCS-Heuristic in terms of energy consumption for all the problem instances. In terms of running time, IOETCS-Heuristic runs slightly faster than IOETCS-ILP.

7 Conclusion

We investigate the problem of energy-aware mapping and scheduling of tasks and communications with conditional precedence constraints and individual deadlines on a heterogeneous NoC-based MPSoC and propose a novel approach. Our approach reduces the total expected energy consumption by collectively optimizing the voltages/frequencies of processors and NoC links. The IOETCS algorithm maps tasks to processors and serializes communications that use same communication links. It constructs a unified schedule and assigns voltages/frequencies to tasks and communications collectively assuming continuous voltages/frequencies. The IOETCS algorithm significantly narrows down the search space for our ILP-based algorithm and our heuristic for assigning discrete frequencies/voltages to tasks and communications. The experimental results show that in terms of energy consumption, our approach using either ILP or heuristic outperforms the state-of-the-art approach proposed by Li and Wu [24] that considers only unconditional task graphs. Compared to the state-of-the-art approach, our ILP-based approach achieves an average improvement of 31%, a maximum improvement of 61% and a minimum improvement of 9%, and our heuristic-based approach achieves an average improvement of 20%, a maximum improvement of 46% and a minimum improvement of 2%. In terms of running time, our approach is approximately 3 times faster than the state-of-the-art approach.

8 References

[1] "Mobile processor exynos 5 octa (5422)," http://www.samsung.com/semiconductor/minisite/Exynos/Solution/MobileProcessor/Exynos_5_Octa_5422.html, accessed: 2017-09-4.

[2] "Standard task graph," URLhttp://www.kasahara.elec.waseda.ac.jp, accessed: 2017-09-4.

[3] "Zynq ultrascale+ mpsocs," https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html, accessed: 2017-09-4.

[4] A. Andrei, P. Eles, Z. Peng, M. T. Schmitz, and B. M. Al Hashimi, "Energy optimization of multiprocessor systems on chip by voltage selection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 262–275, 2007.

[5] T. D. Burd and R. W. Brodersen, "Energy efficient cmos microprocessor design," in *Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, vol. 1. IEEE, 1995, pp. 288–297.

[6] Y. Cai, M. T. Schmitz, B. M. Al-Hashimi, and S. M. Reddy, "Workload-ahead-driven online energy minimization techniques for battery-powered embedded systems with time-constraints," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 1, p. 5, 2007.

[7] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal dvfs and dpm combination," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 3s, p. 111, 2014.

[8] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 24, no. 1, pp. 18–28, 2005.

[9] P. Eles, K. Kuchcinski, Z. Peng, A. Doboli, and P. Pop, "Scheduling of conditional process graphs for the synthesis of embedded systems," in *Proceedings of the conference on Design, automation and test in Europe*. IEEE Computer Society, 1998, pp. 132–139.

[10] M. Engel and O. Spinczyk, "A radical approach to network-on-chip operating systems," in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*. IEEE, 2009, pp. 1–10.

[11] Y. Ge, Y. Zhang, P. Malani, Q. Wu, and Q. Qiu, "Low power task scheduling and mapping for applications with conditional branches on heterogeneous multi-processor system," *Journal of Low Power Electronics*, vol. 8, no. 5, pp. 535–551, 2012.

[12] C. H. Gebotys and R. J. Gebotys, "Power minimization in heterogeneous processing," in *Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences, 1996.*, vol. 1. IEEE, 1996, pp. 330–337.

[13] P. Ghosh, A. Sen, and A. Hall, "Energy efficient application mapping to noc processing elements operating at multiple voltage levels," in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*. IEEE Computer Society, 2009, pp. 80–85.

[14] Q. Gu, P. Lago, H. Muccini, and S. Potenza, "A categorization of green practices used by dutch data centers," *Procedia Computer Science*, vol. 19, pp. 770–776, 2013.

[15] G. Guindani and F. G. Moraes, "Achieving qos in noc-based mpsocs through dynamic frequency scaling," in *2013 International Symposium on System on Chip (SoC)*. IEEE, 2013, pp. 1–6.

[16] B. Guo, J. Yu, B. Liao, D. Yang, and L. Lu, "A green framework for dbms based on energy-aware query optimization and energy-efficient query processing," *Journal of Network and Computer Applications*, vol. 84, pp. 118–130, 2017.

[17] J. Guo and M. Potkonjak, "Coarse-grained learning-based dynamic voltage frequency scaling for video decoding," in *26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2016*. IEEE, 2016, pp. 84–91.

[18] R. Harmon, H. Demirkan, N. Auseklis, and M. Reinoso, "From green computing to sustainable it: Developing a sustainable service orientation," in *43rd Hawaii International Conference on System Sciences (HICSS), 2010*. IEEE, 2010, pp. 1–10.

[19] C. Hasan and Z. J. Haas, "Deadline-aware energy management in data centers," in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2016*. IEEE, 2016, pp. 79–84.

[20] W. Huai, Z. Qian, X. Li, G. Luo, and S. Lu, "Energy aware task scheduling in data centers." *JoWUA*, vol. 4, no. 2, pp. 18–38, 2013.

[21] J. Huang, C. Buckl, A. Raabe, and A. Knoll, "Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems," in *19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2011*. IEEE, 2011, pp. 447–454.

[22] J. In, S. Shirani, and F. Kossentini, "Jpeg compliant efficient progressive image coding," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 1998.*, vol. 5. IEEE, 1998, pp. 2633–2636.

[23] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 3, p. 23, 2007.

[24] D. Li and J. Wu, "Energy-efficient contention-aware application mapping and scheduling on noc-based mpsocs," *Journal of Parallel and Distributed Computing*, vol. 96, pp. 1–11, 2016.

[25] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2015.

[26] M. Lombardi, M. Milano, M. Ruggiero, and L. Benini, "Stochastic allocation and scheduling for conditional task graphs in multi-processor systems-on-chip," *Journal of scheduling*, vol. 13, no. 4, pp. 315–345, 2010.

[27] Z. Lu, "Using wormhole switching for networks on chip: Feasibility analysis and microarchitecture adaptation," Ph.D. dissertation, KTH, 2005.

[28] P. Malani, P. Mukre, Q. Qiu, and Q. Wu, "Adaptive scheduling and voltage scaling for multiprocessor real-time applications with non-deterministic workload," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 652–657.

[29] C. Marcon, N. Calazans, F. Moraes, A. Susin, I. Reis, and F. Hessel, "Exploring noc mapping strategies: an energy and timing aware technique," in *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*. IEEE Computer Society, 2005, pp. 502–507.

[30] S. Mittal, "Power management techniques for data centers: A survey," *arXiv preprint arXiv:1404.6681*, 2014.

[31] ——, "A survey of techniques for improving energy efficiency in embedded computing systems," *International Journal of Computer Aided Engineering and Technology*, vol. 6, no. 4, pp. 440–459, 2014.

[32] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb, "The alpha 21364 network architecture," in *Hot Interconnects 9, 2001*. IEEE, 2001, pp. 113–117.

[33] P. Pop, *Scheduling and communication synthesis for distributed real-time systems*. Department of Computer and Information Science, Linköpings universitet, 2000.

[34] A. Roukh, L. Bellatreche, N. Tziritas, and C. Ordonez, "Energy-aware query processing on a parallel database cluster node," in *Algorithms and Architectures for Parallel Processing*. Springer, 2016, pp. 260–269.

[35] O. Sarood, A. Langer, A. Gupta, and L. Kale, "Maximizing throughput of overprovisioned hpc data centers under a strict power budget," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014, pp. 807–818.

[36] D. Shin and J. Kim, "Power-aware scheduling of conditional task graphs in real-time multiprocessor systems," in *Proceedings of the 2003 international symposium on Low power electronics and design*. ACM, 2003, pp. 408–413.

[37] ——, "Communication power optimization for network-on-chip architectures," *Journal of Low Power Electronics*, vol. 2, no. 2, pp. 165–176, 2006.

[38] J. Singh, S. Betha, B. Mangipudi, and N. Auluck, "Contention aware energy efficient scheduling on heterogeneous multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1251–1264, 2015.

[39] U. U. Tariq and H. Wu, "Energy-aware scheduling of conditional task graphs with deadlines on mpsocs," in *IEEE 34th International Conference on Computer Design (ICCD), 2016*. IEEE, 2016, pp. 265–272.

[40] ——, "Energy-aware scheduling of periodic conditional task graphs on mpsocs," in *Proceedings of the 18th International Conference on Distributed Computing and Networking*. ACM, 2017, p. 13.

[41] D. Wu, B. M. Al-Hashimi, and P. Eles, "Scheduling and mapping of conditional task graph for the synthesis of low power embedded systems," *IEE Proceedings-Computers and Digital Techniques*, vol. 150, no. 5, pp. 262–273, 2003.

[42] Z. Xu, Y.-C. Tu, and X. Wang, "Pet: reducing database energy cost via query optimization," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1954–1957, 2012.

[43] T. T. Ye, G. D. Micheli, and L. Benini, "Analysis of power consumption on switch fabrics in network routers," in *Proceedings of the 39th annual Design Automation Conference*. ACM, 2002, pp. 524–529.

[44] A. N. Yurii Nesterov, *Interior Point Polynomial Algorithms in Convex Programming*. SIAM, 1987.

[45] W. Zhang, E. Bai, H. He, and A. M. Cheng, "Solving energy-aware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms," *Sensors*, vol. 15, no. 6, pp. 13 778–13 804, 2015.